第五課 JavaScript 入門

HTML網頁設計課程

講師：巫宏鈞

Javascript 基礎

- JavaScript 是一種腳本語言
  - JavaScript 是一種輕量級的編程語言。
  - JavaScript 是可插入 HTML 頁面的編程代碼。
  - JavaScript 插入 HTML 頁面後，可由所有的現代瀏覽器執行。
  - JavaScript 很容易學習。

Javascript 基礎

- JavaScript 使用
  - HTML 中的腳本必須位於 <script> 與 </script> 標籤之間。

```
<script>
    alert("Hello world");
</script>
```

Javascript 基礎

- 寫到文檔輸出
  - 下面的例子直接把 <h1> 元素寫到 HTML 文檔輸出中：

```
document.write("<h1>This is a heading</h1>");
```

  - 如果在文檔已完成加載後才執行，則整個 HTML 頁面將被覆蓋：

```
<button onclick="myFunction()">點擊這裡</button>
<script>
    function myFunction() {
        document.write("覆蓋所有內容了！");
    }
</script>
```

Javascript 基礎

- 操作HTML 元素
  - 如需從 JavaScript 訪問某個 HTML 元素，您可以使用 document.getElementById(id) 方法及 innerHEML 屬性。
  - 請使用 id 屬性來標識 HTML 元素：。

```
<p id="demo" >I am demo</p>
<script>
    document.getElementById("demo").innerHTML = "JavaScript";
</script>
```

Javascript 基礎

- JavaScript 註解
  - JavaScript 不會執行註解。所以我們可以利用添加註解來對 JavaScript 進行解釋，或者提高代碼的可讀性。
  - 單行註釋以 // 開頭。
    //我是註解
  - 多行註釋以 /* 開始，以 */ 結尾。
    /* 我是註解1
    我是註解2 */

Javascript 基礎

- JavaScript 除錯
  - 通過 console.log() 可以將欲檢視的內容藉由控制台顯示。

```
<body>
    <script>
        console.log(document.body.clientWidth;);
    </script>
</body>
```

變數

- 變數是存儲信息的容器。
- 就像代數一樣，可以透過變數做資料的運算。

```
var x=2;
var y=3;
var z=x+y;
```

- 變量可以使用短名稱（比如 x 和 y），也可以使用描述性更好的名稱（比如 age, sum, total_volume）。
- 變數必須以字母開頭
- 變數名稱對大小寫敏感（y 和Y 是不同的變量）。

變數的生命週期

- 局部 JavaScript 變數
  - 在 JavaScript 函數內部宣告的變數（使用 var）是局部變數，所以只能在函數內部訪問它。
  - 所以我們可以在不同的函數中使用名稱相同的局部變數，只要函數運行完畢，本地變數就會被刪除。
- 全局JavaScript 變數
  - 在函數外聲明的變數是全局變數，網頁上的所有腳本和函數都能訪問它。
  - 如果把值指定給尚未聲明的變數（未使用 var），該變數將被自動作為全局變數聲明。

函數

- 函數是由事件驅動的或者當它被調用時執行的可重複使用的代碼塊。
- 函數包裹在大括號中的代碼塊，前面使用了關鍵詞 function：

```
function name()  {
    這裡是要執行的代碼
}
```

- 當調用該函數時，會執行函數內的代碼。
- 可以在某事件發生時直接調用函數（比如當用戶點擊按鈕時），並且可由 JavaScript 在任何位置進行調用。

函數

- 操作範例：

```
<html>
<head>
    <script>
    function myFunction() {
        alert("Hello World!");
    }
    </script>
</head>
<body>
    <button onclick="myFunction()">點擊這裡
</button>
</body>
</html>
```

帶參數的函數

- 在調用函數時，您可以向其傳遞值，這些值被稱為參數。而這些參數可以在函數中使用。
- 您可以發送任意多的參數，由逗號(,) 分隔：

```
function myFunction( var1, var2 ) {
    這裡是要執行的代碼
}
```

- 變數和參數必須以一致的順序出現。第一個變數就是第一個被傳遞的參數的給定的值，以此類推。

帶參數的函數

- 操作範例：

```
<script>
function myFunction(name,job) {
    alert("Welcome " + name+ ", the " + job);
}
</script>

<button
onclick="myFunction('Dragon','Teacher')">
    點擊這裡
</button>
```

帶有返回值的函數

- 有時，我們會希望函數將值返回調用它的地方。
- 通過使用 return 語句就可以實現。
- 使用 return 語句時，函數會返回指定的值。：

```
function myFunction() {
    var x=5;
    return x;
}
```

帶有返回值的函數

- 操作範例：

```
<div id="demo"></div>

function myFunction(a, b) {
    return a * b;
}
document.getElementById("demo").innerHT
ML = myFunction(4,3);
```

- 註：由於本範例為直接輸出，故必須將 js 寫於 HTML 元素後，否則會找不到該元素！

物件

- 真實生活中，一輛汽車是一個物件。
- 物件有它的屬性，如重量和顏色等，方法有啟動停止等。

物件	屬性	方法
	car.name = Fiat car.model = 500 car.weight = 850kg car.color = white	car.start() car.drive() car.brake() car.stop()

物件

- 在 JavaScript中，幾乎所有的事物都是物件。
- 以下程式碼為變數 car 設定值為 "Fiat" :

```
var car = "Fiat";
```

- 而物件也是一個變數，但物件可以包含多個值 (多個變數)

```
var car = {
    type : "Fiat",
    model : 500,
    color : "white“
};
```

物件定義

```
var person = {
    firstName: "John",
    lastName : "Doe",
    id : 5566,
    getName : function()
    {
        return this.firstName + " " + this.lastName;
    }
};
```

存取物件屬性與方法

- 存取物件屬性有兩種方式：
  - person.lastName;
  - person["lastName"];
- 存取物件方法：
  - name = person.getName();
- 存取物件方法函數式：
  - name = person.getName;

陣列

- 如果你有一組資料（例如：車名字），存在單獨變數如下所示：

```
var cars1 = "FORD";
var cars2 = "TOYOTA";
var cars3 = "MAZDA";
```

陣列

- 陣列物件的作用就是：
  使用單獨的變數名來儲存一系列的值。

```
var cars = new Array();
cars[0] = "FORD";
cars[1] = "TOYOTA";
cars[2] = "MAZDA";
```

陣列

- 簡潔方式：

```
var cars=new Array("FORD"," TOYOTA ","MAZDA");
```

字面方式：

```
var cars=["FORD"," TOYOTA","MAZDA"];
```

存取陣列

- 存取 cars 陣列的第一個值：

```
var name=cars[0];
```

修改陣列 cars 的第一個元素：

```
cars[0]="BMW";
```

陣列中可以有不同的物件

- 你可以在一個陣列中包含物件元素、函式、陣列：

```
myArray[0]=Date.now;
myArray[1]=person;
myArray[2]=cars;
```

陣列方法和屬性

- 取得陣列中元素的數量：
  var x=cars.length;

- 取得陣列中某個值的索引值：
  var y=cars.indexOf("TOYOTA");

- 用陣列的元素組成字串：
  var z=cars.join();

陣列方法和屬性

- 取得陣列中元素的數量：
  var x=cars.length;

- 取得陣列中某個值的索引值：
  var y=cars.indexOf("TOYOTA");

- 用陣列的元素組成字串：
  var z=cars.join();

陣列方法和屬性

- 將陣列中的元素反轉：
  cars.reverse();

- 將陣列中的元素依字母排序：
  cars.sort();

- 將陣列中的元素依數字排序：
  cars. sort(function(a,b){return a-b});
  cars. sort(function(a,b){return b-a});

陣列方法和屬性

- 將陣列轉為字串並回傳：
  cars.toString();
- 在陣列開頭寫入元素：
  cars.unshift("BMW", "BENZ");
- 在陣列結尾寫入元素：
  cars.push("BMW", "BENZ");

算數運算子

- 算術運算子用於執行變數或值之間的算術運算。

運算符	描述	例子	結果
+	加	x=y+2	x=7
-	減	x=y-2	x=3
*	乘	x=y*2	x=10
/	除	x=y/2	x=2.5
%	求餘數(保留整數)	x=y%2	x=1
++	累加	x=++y	x=6
--	遞減	x=--y	x=4

指定運算子

- 指定運算子用於給 JavaScript 變量指定值。

運算符	例子	等價於	結果
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=xy	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

字串串接運算子

- 可使用 + 運算子把兩個或多個字符串變數連接起來。

```
txt1="What a very";
txt2="nice day";
txt3=txt1+ txt2;
```

在以上語句執行後，變數 **txt3** 的值將是 "What a verynice day"

- 註：如果把數字與字串相加，結果將成為字串

比較運算子

- 比較運算子通常使用在邏輯語句中，以判定變數或值是否相等。

運算子	描述	例子
==	等於	x==8 為false
===	全等（值和類型）	x===5 為true；x==="5" 為false
!=	不等於	x!=8 為true
>	大於	x>8 為false
<	小於	x<8 為true
>=	大於或等於	x>=8 為false
<=	小於或等於	x<=8 為true

判斷式

- 通常在寫代碼時，總是需要為不同的決定來執行不同的動作。我們可以在代碼中使用判斷式來完成該任務。
- 在JavaScript 中，我們可使用以下判斷式：
  - if 判斷式 - 只有當指定條件為 true時執行代碼
  - if...else 判斷式 - 當條件為 true 時執行代碼，當條件為 false 時執行其他代碼
  - if...else if....else 判斷式 - 使用該語句來選擇多個代碼塊之一來執行
  - switch 判斷式 - 使用該語句來選擇多個代碼塊之一來執行。

if 判斷式

```
<p>如果時間早於20:00，會顯示 "Good day"。</p>
<button onclick="myFunction() ">請按這裡</button>
<p id="demo"></p>

<script>
function myFunction() {
    var x="";
    var time=new Date().getHours();
    if (time < 11)  {
        x="Good day";
    }
    document.getElementById("demo").innerHTML=x;
}
</script>
```

If...else 判斷式

```
if (time < 20)  {
    x="Good day";
} else {
    x="Good evening";
}
```

If...else if...else 判斷式

```
if (time<10) {
    x="Good morning";
} else if (time<20) {
    x="Good day";
} else {
    x="Good evening";
}
```

switch 判斷式

```
var day = new Date().getDay();
switch (day) {
    case 0:
        x = "星期一，猴子穿新衣";
        break;
    case 1:
        x = "星期二，猴子肚子餓";
        break;
    case 2:
        x = "星期三，猴子去爬山";
        break;
    default:
        x = "猴子放假去！";
}
```

迴圈

- 如果您希望一遍又一遍地運行相同的代碼，並且每次的值都不同，那麼使用迴圈是很方便的。
- JavaScript 支援不同類型的循環：
  - for - 計次迴圈
  - for/in - 循環陣列迴圈
  - while - 條件式迴圈 (不一定執行)
  - do/while - 條件式迴圈 (一定會執行一次)

for 迴圈

```
for (var i=0; i<5; i++) {
    x = x + "The number is " + i + "<br>";
}
```

- 搭配陣列使用

```
cars=["BMW","Volvo","Saab","Ford"];
for (var i=0;  i<cars.length;  i++) {
    document.write(cars[i] + "<br>");
}
```

for/in 迴圈

```
var x;
var txt = "";
var person={fname:"Bill",lname:"Gates",age:56};
for (x in person) {
    txt = txt + " " + person[x];
}
document.getElementById("demo").innerHTML=text;
```

while 迴圈

```
var x = "", i = 0;
while (i<5) {
    x = x + "The number is " + i + "<br>";
    i++;
}
document.getElementById("demo").innerHTML=x;
```

do/while 迴圈

```
var x = "", i = 0;
do {
    x = x + "The number is " + i + "<br>";
    i++;
} while (i<5);
document.getElementById("demo").innerHTML=x;
```

Break 語句

- break 語句可用於跳出迴圈，並會繼續執行該迴圈之後的代碼（如果有的話）：

```
for (i=0;i<10;i++) {
    if (i==3) {
        break;
    }
    x = x + "The number is " + i + "<br>";
}
document.getElementById("demo").innerHTML
=x;
```

Continue 語句

- continue 語句可以中斷迴圈中的本次代碼，如果出現了指定的條件，然後繼續迴圈中的下一次代碼。

```
for (i=0;i<=10;i++) {
    if (i==3) {
        continue;
    }
    x = x + "The number is " + i + "<br>";
}
document.getElementById("demo").innerHTML
=x;
```

typeof 運算子

- 可以使用 typeof 運算子來檢測變數的資料型態。

```
typeof "John"                    // 回傳 string
typeof 3.14                      // 回傳 number
typeof false                     // 回傳 boolean
typeof [1,2,3,4]                 // 回傳 object
typeof {name:'John', age:34}     // 回傳 object
```

Math 物件

- 取得圓周率
  - Math.PI
- 四捨五入至整數
  - Math.round(值)
- 取得亂數
  - Math.random()
- 無條件進位
  - Math.ceil(值)
- 無條件捨去
  - Math.floor(值)

Date 物件

- 取得當日日期
  - var d = new Date()
- 取得年份
  - d.getFullYear()
- 取得月份
  - d.getMonth()+1
- 取得日
  - d.getDate()
- 取得星期幾
  - d.getDay()
- 設定日期
  - var d = new Date(年,月,日,時,分,秒,豪秒)